Nested Graph Dissection and Approximation Algorithms

Sudipto Guha*
Stanford University, Stanford, CA 94305.

December 1, 2000

Abstract

This paper considers approximation algorithms for graph completion problems using the nested dissection paradigm. Given a super-additive function of interest (the smallest planar or chordal extension for example) and a test that relates it to an upper bound of the smallest separator, we provide a framework how to dissect the graph recursively such that no subgraph has more than half the value of its parent, (or is indistinguishable via separator tests) in polynomial time. Interestingly we cannot bound such a function till we have constructed the entire nested dissection. We achieve a partition of the graph with respect to a constant number of such unknown estimator functions simultaneously.

Using the framework the paper presents improvements in approximating the chordal completion size (by a factor of $\log n$), operation count (by a factor of $\log^2 n$ and the polynomial term depending on degree) and elimination height.

We show that there exists a nested dissection ordering that simultaneously minimizes the elimination height, chordal completion, operation count to within $O(\log n)$ factors of the best possible (which may be obtained by three independent orderings) improving the previous existence theorem by factors of $\log n$ and $d^{\frac{1}{3}} \log^3 n$ for the latter two.

We also show that graphs with small crossing number or fill-in have better approximations of the elimination height, completion and operation count. As a consequence we can approximate the pathwidth, cutwidth, vertex ranking problems better for such graphs.

The paper also improves, in some cases, the approximation results of minimum drawing size (number of vertices plus the crossing number) of a planar embedding of a graph, and its layout area on a grid.

*`sudipto@cs.stanford.edu` Research Supported by IBM Cooperative Fellowship, NSF Grant IIS-9811904 and NSF Award CCR-9357849, with matching funds from IBM, Mitsubishi, Schlumberger Foundation, Shell Foundation, and Xerox Corporation.

1 Introduction

Graph completion problems arise in various contexts in optimizing computations involving sparse matrices and VLSI layout problems. These problems usually seek the smallest supergraph (adding vertices or edges) that lie in some structured family, interval, planar or chordal for example.

Chordal Completion: One of the most common problems in sparse matrix computation is finding a vector x satisfying the linear system $Ax = b$. It is difficult to overstate how ubiquitous this problem is, see [10] for a survey on its applications. One method of solving this is via Gaussian elimination process, in which a multiple of a row is subtracted from other rows. Parter in [47] interpreted the elimination process of rows as a graph theoretic game, Rose in [49] demonstrated the connection of this problem with finding chordal graphs. (See Appendix A for details, and [1] for a survey of the results.) This problem also arises in context of pedigree analysis, evidence propagation in belief networks, see [29].

A graph is defined to be *strongly chordal* or just *chordal*, if there exists no induced simple chordless cycle of length 4 or more. In *the minimum chordal completion problem* we are required to construct a chordal graph with the fewest edges. This bounds the space required to solve a sparse linear system. This problem has received considerable attention from researchers; see [9, 15, 17, 18, 19, 21, 22, 25, 29, 37, 36, 41, 42, 50, 54, 57], amongst many others. Most of these approaches use one of two methods, minimum degree heuristic (see [37, 42, 54] for example) or nested dissection (see [17, 19, 36]). Hybrid algorithms (see [2, 27]), are also used.

The space requirement is not the only measure of interest in solving sparse linear systems. The total time taken, which is the number of arithmetic operations or *the operation count* is an useful measure if we have to solve the fixed system for different inputs b. The minimum number of rounds required if rows can be eliminated in parallel, the *the elimination height*, is also important in this context. The problem of minimizing operation count has not been studied as extensively as the problem of minimizing fill-in. Some results are shown in [26, 36, 25]. Minimizing the elimination height has been studied with regard to eliminating more than one row in parallel. The elimination tree was first defined in [51], although several algorithms prior to this paper used the idea. A considerable number of other heuristics have been proposed for this problem as well, see [15, 28, 33, 34, 39, 40]. [38] is an excellent survey on uses of elimination trees. In [28, 39] the smallest height elimination tree was constructed for chordal graphs.

All of the above problems are NP–Hard. In [1, 30] the first polynomial time approximation for these were proposed. This was improved and supplemented in [44]. In [1] it was shown that there exists an ordering which simultaneously approximates the three most known parameters, approximation results were also provided. In our work we improve both the existential and the approximability results. In [7] it was investigated if there exist better orderings for chordal graphs to expose parallelism, increasing the fill or work required by a small amount.

Interval Graph Completion: The *Pathwidth* of a graph is defined in the following section. The cutwidth of a graph is the maximum number of edges that would be cut if the vertices of the graph are mapped to a path. The problems of minimizing pathwidth or cutwidth of a graph are interesting in some VLSI gate layout problems, see [43, 52]. The problem also appears in minimizing vertex or edge rankings of graphs which are used in manufacturing systems [45], (see [6] for more details). In [30, 1, 5] an $O(\log^2 n)$ approximation was presented.

Planar Completion: The *Minimum Drawing Size* is defined as : given a graph G, to provide a planar embedding mapping vertices to points and edges to simple curves, such that the total number of points are minimized. At most two edges can intersect at a *crossing* point other than a vertex. This is a very natural problem in graph drawing and planar layouts. VLSI layout problems also fall into this category, since for most such problems a graph of degree at most 4 is to be embedded in a grid. There are several possible measures to capture desirability of an embedding. One natural measure, the area, given that no two edges of the graph which are mapped to paths on the grid share a link, has received considerable attention in [32, 56, 3, 55, 53, 31]. In [14] the results on minimizing planar graph completion and minimizing area of VLSI layouts were improved.

Nested Dissection: One of the most celebrated techniques to solve these completion problems is divide and conquer. The initial step is to construct separator trees, and next construct the completion of the two subgraphs and subsequently of their union. The construction of the separator tree is referred to as nested dissection. This divide and conquer algorithm was first proposed in [17], and followed extensively. In [32] the problem of minimum area VLSI layout for planar graphs was addressed by the same technique. This approach has been followed in [8, 31, 1, 30].

All known approaches require the graph to be constant degree or planar to bound the interaction in the conquer step. Another key ingredient is the existence of small separators for these families of graphs. Also the graph is expected to be divided into roughly equal parts to bound the depth of the recursion. This is natural since in each level we will introduce errors and reducing the number of levels will help.

The following interesting problem appears in this context, that the separators are chosen *before* the graph is constructed, and typically the interaction of these separators determines the goodness of the final solution. So if we had a way of constructing separators with foresight as to how it would interact with the *separators we are going to construct later*, we would have completion graphs with better qualities. The natural question that arises is *what is the best possible way to partition the graph ?* It is immediate that as the 'first cut' we are better off partitioning the graph with respect to its size. As we progress, we will discover that the contribution to the objective function is more from one part than the other. At this point the most natural response would be to 'back up' a few steps and try partitioning the larger side, hoping we get it right. This process actually means that we will try several different cuts at each level and possibly backtrack as well. In this scenario it is not at all immediate how to achieve all this in polynomial time.

This idea was explored in [14] in context of minimizing crossing number and its subsequent application in VLSI layouts. In this paper we develop the idea and show that the methods of [14] extend to a general framework. The framework starts from the observation that dividing the graph according to the *final measure of quality* is better for the conquer step. Although we cannot compute this measure before making the divisions, we can achieve this goal provided the measure of quality we use satisfies some very simple properties. (Of course we would still use the small separator property.) The measure should be super additive, that is for any two disjoint vertex induced subgraphs, the sum of their measure should be less than the measure of the whole graph. We will define this property more formally later. We will see that simple measures like number of edges and vertices will satisfy this requirement.

One technical difference from [14] is in using vertex separators instead of edge separators. The edge separators will not always yield improvements, for computing chordal completion of graphs (see [25] in context of planar graphs) and will give polynomially worse guarantees. This makes the description of the framework simpler as well as more difficult at the same time. In the end we relax the assumptions required for the decomposition and simplify the proofs considerably. We will be omitting most of the proofs due to lack of space. However the proof of the general framework is left in place to convince the reader of its simplicity.

The benefit of investigating and establishing such a framework is that subsequent applications of the framework yields approximation results for graph completion problems very easily. The problems reduce to choosing a suitable function to partition the graph on. Viewing the graph dissection as an abstract problem also allows us to deal with multiple objective functions.

Our Results: We present a $O(\sqrt{d}\log^3 n)$ approximation for the minimum chordal completion problem and $O(d^{\frac{2}{3}}\log^4 n)$ approximation for the operation count for graphs with bounded degree d. In effect, this is the first $o(n)$ approximation for the operation count problem, since we improve the dependence on the parameter d. We improve the approximation ratio of these quantities by factors of $O(\log n)$ and $O(d^{\frac{1}{3}}\log^2 n)$ respectively. Combined with results in [44], this improves the best approximation ratio for the minimum fill-in problem by a logarithmic factor. We present a different way of accounting for fill-in and operation count than in [1, 30] which allows the improved results to be proved.

We show that there exists a nested dissection ordering that simultaneously minimizes the elimination height, chordal completion, operation count to within $O(\log n)$ factors of the best possible (which may be obtained by three independent orderings). The results for the last two hide $\sqrt{d}$ and $d^{\frac{2}{3}}$ factors, but improves the previous results in [1, 30] by factors of $\log n$ and $d^{\frac{1}{3}}\log^3 n$. We show for a graph with fill-in F, we can simultaneously approximate the elimination height within $O(\log n \log F)$, the completion size within $O(\sqrt{d}\log^2 n \log F)$ and the operation count within $O(d^{\frac{2}{3}}\log^3 n \log F)$. Based on this technique we can approximate the pathwidth and cutwidth of a graph better, if the graph has small crossing number or fill-in. We also prove some small results in context of minimizing the planar completion.

2 Preliminaries, Definitions and Notation

Definition 1 *An* **Elimination Ordering** *of the vertices is a mapping from the vertices to* $1 \ldots |V|$*. The vertices are removed in increasing order (of the map) and when vertex i is removed, all (remaining) neighbors of i are joined by a clique. This process is defined as the* **Elimination Game**.

Definition 2 *The* **Elimination Tree** *is the tree formed by an elimination ordering where the parent of i is the smallest numbered neighbor of i when i is removed. The* **Elimination Height** *is the height of this tree.*

Definition 3 *The* **Fill-in** *is defined to be the* **additional edges** *added by an elimination game.*

Definition 4 *The* **Chordal Completion Problem** *is to add minimum number of edges to a given graph such that the final graph is chordal. It is also referred to as triangulating a graph.*

Definition 5 *The* **Operation Count** *is the number of arithmetic operations required to reduce a Gaussian System. Given an elimination game, the operation count is the sum (over i) of the square of the size of the neighborhood of i which is still present in the graph when i is removed.*

Definition 6 *A* **Path Decomposition** *of a graph is a collection of subsets X_i of the graph such that for each edge (u,v), they coexist in a subset, and for all $i < j < k$ we have $X_i \cap X_k \subseteq X_j$. The* Width *of this decomposition is one less the size of the largest subset. The* Pathwidth *is defined to be the minimum width over all path decompositions.*

Definition 7 *A* **separator** *is a vertex on whose deletion the vertex set gets split such that no part has more than $2/3$ of the original number of vertices.*

Definition 8 *A* **separator tree** *is defined to be a recursive decomposition using separators. We will abuse the notation and use separator trees to mean recursive decomposition using vertex cuts.*[1]

Throughout this paper, we will use G to denote a graph, F to denote the minimum fill-in, H the minimum elimination height. The symbol $\phi(G)$ will indicate the size of the minimum chordal completion of the graph G. The operation count will be indicated by $OPT_m(G)$. Most graphs considered here will be of bounded degree, where d will denote the maximum degree. The symbols n and m will indicate the number of vertices and edges of the input graph.

3 Separator Trees with respect to unknown weights

Consider a special family $\mathcal{F}$ of graphs which is closed under induced subgraphs. Assume that a given graph G can be extended (possibly with modification to its topology, planar embeddings for example) to belong to this family. It is possible that a graph can be extended to more than one member, which naturally poses the question of "small" extension. Assume there exists a function of which maps the graphs of $\mathcal{F}$ to some integer.

Definition 9 *We define the function value of the minimum extension of the graph G as an* **estimator** *of G. We will refer to the function as estimator function henceforth.*

Consider a *Graph Dissection Problem* in which a given graph G is to be recursively decomposed with respect to the estimator function (or the minimum extension).[2] Clearly for arbitrarily behaved estimator functions this may not be be feasible. We introduce the following definition,

Definition 10 *An estimator function is ϕ defined to be* **super-additive** *if for an arbitrary vertex induced subgraph P*

1. *ϕ is Monotone, $\phi(P) \leq \phi(G)$.*
2. *ϕ is super-additive, that is $\phi(P) + \phi(G - P) \leq \phi(G)$.* [3]
3. *There exists a function $f()$ which provides an upper bound for a separator for this graph with respect to an arbitrary subset of vertices. This function depends on $\phi(G)$ and possibly on number of vertices or edges or other parameters of the graph G.*

We observe that the crossing number of a graph is a super-additive estimator, and so is the minimum fill-in number. Assume that $\alpha(G)$ approximation for the separator can be found in polytime. For most of the cases we would be concerned with here, the estimator of the extension of the given graph G is polynomially bounded in the number of vertices of G. Otherwise the $\log n$ terms in the various bounds we will achieve will be replaced by $\log \phi$. we will mostly use the Leighton-Rao approximation [31] of the minimum separator. For some special cases we will consider other separator algorithms. Example of such separator algorithms are 2 approximation for planar separators [16], and approximation for graphs of bounded genus [13] etc. We will use the following easy claim throughout,

Claim 3.0.1 *If we find a separator of size at most s that separates an arbitrary set, then we can find a separator of size $3s$ that simultaneously separates the entire vertex set and another specified subset. (The definition of separation has to be relaxed in the simultaneous context to mean some constant fraction separation instead of $1/3, 2/3$.)*

[1] The cuts separate the vertex set with respect to some weight function, not necessarily the size.

[2] In reality we will almost always be concerned with construction of a small extension of the graph, and use the recursive decomposition as an initial step to construct the extension.

[3] Actually it will be sufficient that $\phi(P) + \phi(G - P) \leq (2 - \delta)\phi(G)$ where δ is bounded away from zero.

The claim follows from first applying it with respect to the vertex set and then applying it to the subsets of the specified subset in the two pieces constructed, see [14].

3.1 The Main Dissection Theorem

Theorem 3.1 *For a super-additive estimator ϕ, in polynomial time a decomposition tree can be constructed where the parameter t at the root of a subtree (corresponding to a subgraph G') decreases by a factor of $1/2$ along a path, and the separator corresponding node is at most $O(\alpha(G')f(t,G'))$.* [4]

Consider we are attempting to partition a graph G with respect to the tentative value t for the estimator $\phi(G)$, denoted by $\textsc{Partition}(t,G)$. To partition the graph we will call $\textsc{Partition}(M,G)$, where M is a large enough value greater than $\phi(G)$. We will assume throughout the rest of this article that $\log M = O(\log n)$. As we will shortly see, all we would need is that M be large enough, the algorithm will 'correct' the value of M within a factor of 2 of $\phi(G)$. Returning to the algorithm,

Algorithm $\textsc{Partition}(t,G)$

1. We assume that if t is a small constant, we can construct an appropriate decomposition tree.
2. Initially the set $X_0 = \Phi$, and $N_0 = G$, and number of iterations $i = 0$ so far. N_i will denote the graph being considered at the i'th iteration with the vertex set V_i.
3. Recursively for iteration $i+1$,
   (a) Find an approximate simultaneous balanced separator S_i of N_i with respect to the vertex set X_i and V_i.
   (b) If $S_i \geq c_\alpha \alpha(N_i) f(t,N_i)$, then clearly the estimator is incorrect. We report failure for $\textsc{Partition}(t,G)$.
   (c) Let the two subgraphs created be A and B. We now recursively try the procedures $\textsc{Partition}(t/2,A)$ and $\textsc{Partition}(t/2,B)$.
   (d) Both return failure, we report failure to $\textsc{Partition}(t,G)$.
   (e) One of them returns success, say A. Make $N_{i+1} = B \cup S_i$, set $X_{i+1} = S_i \bigcup (B \bigcap X_i)$ and proceed to $i+1$'th iteration.
   (f) Both return success,
      i. We try $\textsc{Partition}(t/2, G - N_i)$. (If $i = 0$, the top three steps are redundant.)
      ii. If it returns failure, we return failure to our original function $\textsc{Partition}(t,G)$.
      iii. We take the three decompositions of A, B, and $G - N_i$, and the common separator is $S_i \cup X_i$. (*See section 3.2 below.*)
      iv. At this point we have a decomposition for $\textsc{Partition}(t,G)$. However this decomposition is possibly bad because t was an inflated estimator. We now call $\textsc{Partition}(t/2,G)$. If this returns failure, we return the decomposition tree constructed above.
      v. Otherwise we return the decomposition returned by $\textsc{Partition}(t/2,G)$, notice that the way the recursion is defined, this decomposition could have been constructed for a much smaller value of the estimator.

[4] In case we have $\phi(P) + \phi(G-P) \leq (2-\delta)\phi(G)$ (See footnote 3.) the algorithm can be adapted that $\phi()$ decreases by $1 - \delta/2$ along a path.

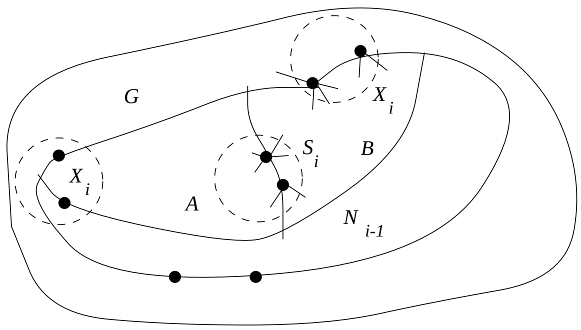

Figure 1: The sets N_i etc. ...

The algorithm is best explained by the figure 1, the set X_i is shown over in two parts, the set S_i separates A and B. So at any point of time the set X_i separates N_i from the rest of the graph G. The final separator $S_i \cup X_i$ separates the graph into three pieces, A, B and $G - N_i$. All vertices but one of X_i shown are also in X_{i-1}. Let us now prove the following claims,

Lemma 3.1 *If $t \geq \phi(G)$ then the above algorithm returns a decomposition successfully.*

Proof: We will prove this lemma by induction on all subgraphs $V(G)$.

Let us consider the points where failure is returned. Assume that the failure is returned in the $i+1$th iteration. Clearly from the assumptions that the estimator function is non-increasing on subgraphs, we have $\phi(N_i) \leq t$ and $S_i \leq \alpha(N_i) f(\phi(N_i), N_i)$. Also if both the parts A and B return failure, then by induction hypotheses we have $\phi(N_i) \geq t$, which means $\phi(G) > t$.

Thus the only case left to consider is the case when both A and B were successfully partitioned, but since there was a call to subgraph of N_i (in the iteration trying to split N_{i-1}, removing the vertices of S_{i-1}) which returned failure at $t/2$. Therefore $\phi(N_i) > t/2$. So if Partition$(t/2, G - N_i)$ returns failure, by super-additivity, $\phi(G) > t$. □

Lemma 3.2 *The final separator constructed is at most $O(\alpha(G) f(t, G))$.*

Proof: It suffices to prove that the sets X_i are all bounded by $O(\alpha(G) f(t, G))$. Since $|X_0| = 0$, and the following is obeyed for some $a < 1$, which is the separation constant as in claim 3.0.1,

$$|X_i| \leq a|X_{i-1}| + |S_{i-1}| \leq a|X_{i-1}| + c_\alpha \alpha(G) f(t, G)$$

where c_α is the constant in the separator test, the lemma follows immediately. □

Therefore theorem 3.1 is true by inspection, since in this decomposition tree the upper bound of the ϕ decreases by a factor of 1/2. However we still need to prove that the above algorithm runs in polynomial time, which is shown in the subsequent lemma.

Lemma 3.3 *The algorithm* Partition *runs in polynomial time.*

Proof: Let $g(t, n)$ denote the running time of Partition(t, G) where the size of G is n. Let the nonrecursive part in the above procedure take time $p(n)$. Then if we inspect the recursive structure of the program, the recurrence can be written as follows,

$$g(t, n) \geq p(n) + g(t/2, (1-a)n) + \sum_{i=1} g(t/2, a^i n) + g(t/2, n) + \sum_{i=1} g(t/2^i, n)$$

where the separation with respect to vertex set V_i introduces no piece larger than $a|V_i|$, clearly $1/2 \leq a < 1$. We assume $p(n) = n^\gamma$, for some constant γ. The above can be shown to have a solution which is $g(t,n) = cn^\gamma t^\delta$, where c and δ are constants which satisfy

$$c \geq 1 + \frac{c}{2^\delta}\left[(1-a)^\gamma + \frac{a^\gamma}{1-a^\gamma} + \frac{2-1/2^\delta}{1-1/2^\delta}\right]$$

□

3.2 Two way or Three way

In the above we showed three way decomposition. For most of the application we can think of, this appears sufficient, however the following can be claimed if we want *two way* decomposition instead. This is more along lines of the algorithm in [14], with the constants slightly optimized. See Appendix Section B for more discussion. For the rest of this paper we will simply *ignore* issues of two way or three way decomposition. All results on three way decomposition , as in the next section, will carry over to the two way case.

3.3 Nested Dissection with Several Estimators

We can extend the algorithm to create a dissection with respect to a constant number of estimator functions. Notice that these estimators may not be dependent, in the sense that an extension of a graph that minimizes one estimator function can be poor in terms of another. Surprisingly the decomposition can still be performed. See Appendix Section C for details.

Theorem 3.2 *If $\phi_1, \phi_2, \ldots, \phi_p$ are super-additive estimators with functions $f_1, f_2, \ldots, f_p$ which relate them to the separator size with respect to arbitrary subsets, in polynomial time a decomposition tree can be constructed such that the separator for G' has size at most $\alpha(G')f_j(s_j, G')$ for all $j \in \{1, \ldots, p\}$ and subgraphs G' induced by any subtree of the separator tree. Furthermore in every p'th level starting from level j, the guess s_j for the estimator ϕ_j decreases by a factor of $1/2$ along any path in this tree.*

4 Minimum Chordal Extension and Fill-in

Lemma 4.1 *([24]) Given a chordal graph G with m edges and a set S, there exists a separator for the vertex set S of size $O(\sqrt{m})$.*

We construct the decomposition tree where our estimator is the minimum number of edges in a chordal extension of the graph. This yields a $O(\log n)$ depth tree, since the number of edges is at most n^2. Consider the elimination ordering imposed by the post order traversal of this tree. The vertices within a single node of the separator tree are placed in arbitrary order. We denote an ordering to be a nested dissection ordering if it arises from a separator tree. The ordering defines a function $\beta : V(G) \rightarrow [1..|V(G)|]$. Consider the following lemma,

Lemma 4.2 *([23, 1]) Let $\beta()$ be a nested dissection ordering of G, and $w, v \in G$ such that $\beta(w) \leq \beta(v)$. Let X_i denote the separator node in which i is present. An edge (w, v) is in the chordal completion of G with respect to this ordering only if there exists an edge $(u, v) \in G$ such that X_v is an ancestor of X_w and X_w is an ancestor of X_u.*

At this point we are ready to compute the number of edges required by our ordering. We will proceed along a different accounting scheme than in [1]. We charge an edge (w, v) to the node which is closer to the root node of the separator tree. Let X_v denote the separator containing the node v, let G_v denote the subgraph induced by all vertices in the subtree rooted under v. Since v can have at most d edges, for any edge (w, v) to be present, the vertex w must be one of these d paths. A very naive counting shows that the number of such vertices can be d times the maximum possible number of vertices along any path from the node X_v to a leaf node in the separator tree. Since the estimator decreases by a factor of $1/2$ along any path, the number of such vertices are at most $(2+\sqrt{2})\sqrt{2\phi(G_v)}\log n$. Thus the contribution from all nodes in X_v is at most

$$d \cdot |X_v| \cdot (2+\sqrt{2})\sqrt{2\phi(G_v)}\log n \leq (2+\sqrt{2})d(2\phi(G_v))\log^2 n$$

Summing over all separator nodes at the same level as X_v, their contribution is $O(d(\log^2 n)\sum\phi(G_v))$. Since the sum of $\phi(G_v)$ is made over disjoint subgraphs, by super-additivity of ϕ the contribution from all nodes in a level of the separator tree is at most $O(d(\log^2 n)\phi(G))$. Summing over the $O(\log n)$ levels of the separator tree, this gives a $O(d\log^3 n)$ approximation to the minimum chordal completion problem. We can however be more careful in the accounting. The d paths would share a lot of nodes on the separator tree. Let us first prove a general lemma, (See Appendix 4 for Proof)

Lemma 4.3 *Given a tree with weights on vertices, such that the weights along any path from root to a leaf decrease by a factor of $\gamma < 1$ and for every anti-chain of size s the inequality $\sum_i y_i \leq s^a W$ holds, where W is the weight of the root, and a, γ are constants strictly less than 1; then the sum of the vertex weights along any r paths can be bounded by $O(r^a W)$.*

Now in this case $\gamma = \sqrt{1/2}$ and $a = 1/2$. This follows since the estimators along an anti-chain, due to the super-additivity have the property that $\sum_i \phi(G_i) \leq \phi(G)$. The weights on the nodes we considered in lemma 4.3 are $c\sqrt{\phi G_i}\log n$ for some constant c. Thus the number of neighbors of v below v in the separator tree are at most $O(\sqrt{d\phi(G')}\log n$, where G' is the subgraph induced by all nodes which are at the same node or descendent of the node v. Thus summing over all nodes that occur with v, the sum is $O(\sqrt{d}\phi(G')\log^2 n)$. Across all the separators in the same level in the tree, the contribution is $O(\sqrt{d}\phi(G)\log^2 n)$, and summing over the levels gives us the following result,

Theorem 4.1 *The minimum chordal completion problem can be approximated within a factor of $O(\sqrt{d}\log^3 n)$.*

We now use the theorem if [44], that an $\rho(n)$ approximation for the minimum triangulation size implies a $\rho(|OPT|d)d^2$ approximation for the minimum fill-in problem.

Corollary 4.2 *The minimum fill-in problem for graphs of degree at most d, can be approximated within a factor of $O(d^{2.5}\log^3(|OPT|d))$ which is at most $O(d^{2.5}\log^3 n)$.*

4.1 Planar Graphs

Djidjev in [12] showed given planar graph with a cost function on the vertices, we can find a vertex separator of cost $\sqrt{\sum_v cost^2(v)}$. If we set $cost(v) = \sqrt{d_v}$, then this is a separator of size at most $\sqrt{n_i}$. (Since every vertex has cost at least 1.) In the above analysis the contribution of a vertex is $\sqrt{d}$ times $O(\sqrt{n_i})$, using the geometric decrease property. Thus the contribution from the separator the vertex is in, is $O(\sqrt{n_i})$ times the *cost of the separator* (sum of square roots of degrees), which is also $O(\sqrt{(}n_i))$. This reproves the $O(n\log n)$ fill-in for planar graphs in [25].

4.2 Arbitrary Degrees

We can also improve the result for minimum chordal completion for graphs with unbounded degree. An $O(m^{\frac{1}{2}} log^{\frac{7}{2}}/|OPT|^{\frac{1}{4}})$ approximation was presented for this case in [1]. In Appendix D we show,

Theorem 4.3 *The above algorithm gives an $O(m^{\frac{1}{2}} log^{\frac{3}{2}} n/\phi(G)^{\frac{1}{4}})$ approximation for the chordal completion for arbitrary graphs.*

5 Minimum Operation Count

In [1] it was shown that the number of operations can be approximated up to a factor of $O(d \log^6 n)$. Let us denote the minimum operation count required by a graph G by $OPT_m(G)$. We show,

Lemma 5.1 *The minimum operation count, OPT_m, is a special super-additive estimator.*

Lemma 5.2 *If the minimum number of multiplications required by a graph is OPT_m, then for any arbitrary set S there exists a separator of size $O(\sqrt[3]{OPT_m})$.*

Every operation can be charged to an edge, and we will count all the charges associated with an edge at the vertex which is closer to the root of the separator tree.

Lemma 5.3 *Consider a node v and its associated node X_v in the separator tree, and the graph induced by all vertices in the descendent nodes of X_v be G'. The charge at v is at most $O(d^{2/3} OPT_m(G') \log^3 n)$.*

Once again, using the fact that the subtrees at the same level induce disjoint digraphs, we get a total charge of $O(d^{2/3} OPT_m(G) \log^3 n)$ at each level. Summing over all the levels gives the following,

Theorem 5.1 *The minimum number of operations can be approximated within $O(d^{2/3} \log^4 n)$.*

6 Minimum Elimination Height

Minimizing fill-in does not minimize elimination height and vice versa. It is easy to observe the fact for the case of a chain. However we show that graphs with small fill-in have small elimination height.

Lemma 6.1 *([6]) The elimination height for chordal graphs can be approximated for chordal graphs up to a factor of $O(\log n)$.*

Lemma 6.2 *If the elimination height of a graph is H, then by deleting H vertices we can bisect an arbitrary set in the graph, such that each part has at most $2/3$ fraction of the vertices.*

We guess the elimination height H up to a constant.[5] We use the minimum fill-in as the estimator. Our function $f()$ is a constant, $2H$. When the estimator function is 1, we remove two vertices and check if the graph is chordal. If the graph is indeed chordal, we perform a balanced (with respect to nodes) decomposition through clique separators. Otherwise we return failure.

If F is the minimum fill-in of the original graph then after $O(\log F)$ levels, all subgraphs become chordal. We use a $O(\log n)$ approximation for the separator in each level. Thus the height over this part is $O(H \log n \log F)$. Once the subgraphs are chordal, the elimination height is at most $O(H \log n)$

Theorem 6.1 *The minimum elimination height of of a graph can be approximated within a factor of $O(\log n \log F)$ where F is the minimum fill-in required for the graph.*

Theorem 6.2 *The pathwidth (and cutwidth) of a graph can be approximated within $O(\log n \log F)$ for constant degree graphs where the minimum fill-in is F.*

[5] We can also guess it exactly, which will increase the running time by n instead of $\log n$.

7 Simultaneous Approximation

Gilbert in [23] showed that there exits a nested dissection ordering which achieves triangulation size which is $O(\log n)$ times optimal . Agarwal, Klein and Ravi in [1] showed that there exists a nested dissection ordering which gives $O(\log n)$ approximation of the minimum height, $O(\sqrt{d}\log^2 n)$ approximation in the minimum triangulation size and $O(d\log^4 n)$ approximation of the minimum number of operations. The paper also provided an $O(\log^2 n)$, $O(\sqrt{d}\log^4 n)$ and $O(d\log^6 n)$ simultaneous approximations in polynomial time to the three parameters respectively. In this section we improve both the existential and the constructive results. (The proofs can be found in Section G.)

Theorem 7.1 *If the minimum elimination height is H, minimum fill-in F (not necessarily achieved by the same ordering) and minimum number of operations M, there exists a nested dissection ordering which has height $O(H\log n)$, fill $O(\sqrt{d}(m+F)\log n)$ and number of operations $O(d^{2/3}M\log n)$.*

Theorem 7.2 *If the minimum elimination height is H, minimum fill-in F and minimum number of operations M, then in polynomial time we can construct a nested dissection ordering which has height $O(H\log n\log F)$, fill-in $O(\sqrt{d}(m+F)\log^2 n\log F)$ and number of operations $O(d^{2/3}M\log^3 n\log F)$.*

Theorem 7.2 shows that for a chordal graph there exists a nested dissection ordering which achieves elimination height of $O(\log n)$ of best possible while introducing $O(\sqrt{d}m\log n)$ fill and approximating the best operation count by an $O(d^{\frac{2}{3}}\log n)$ factor. It is interesting if the factors of d can be removed. In [7] it was shown that the elimination height of a chordal graph can be approximated within $O(\log^2 n)$ of the best possible while introducing fill-in at most $O(m)$ and making the operation count at most $O(1)$ of best possible.

7.1 Crossing Number in Chordal Completion

We show an interesting connection between between almost planar graphs and chordal completion. The motivation of such a theorem comes from the fact that planar graphs have chordal completion of size $O(n\log n)$. (See Section I for details.)

Theorem 7.3 *The Theorem 7.2 holds with the crossing number k replacing the parameter F. Therefore the pathwidth of a graph can be approximated upto factor $O(\log n\log k)$.*

8 Planar Completion Problems

Using the framework we can show the following which is an improvement over [14] in case of small crossing numbers. (See Section H for details.)

Theorem 8.1 *The minimum planar graph completion problem can be approximated within a factor of $O(\log^2 n\log k)$ or a factor of $O(\max\{k\log k,\log n\})$, where k is the crossing number.*

Theorem 8.2 *The minimum area VLSI layout problem can be approximated within $O(\max\{\log^2 n, k\log k\})$.*

Acknowledgements

We would like to thank Baruch Schieber, Joseph Naor and Sanjeev Khanna for numerous suggestions.

References

[1] A. Agarwal, P. Klein, and R. Ravi. “Cutting Down on Fill using Nested Dissection : Provably Good Elimination Orderings”. In A. George, J. R. Gilbert, J. W. H. Liu, editors, *Graph Theory and Sparse Matrix Computation*, Vol 56 of the IMA Volumes in Mathematics and its Applications, pages 31-55, Springer Verlag (1993).

[2] C. Ashcraft, and J. Liu. “Robust ordering of sparse matrices using multisection”. Technical report, ISSTECH-96-002, Boeing information and support services, (1996).

[3] S. N. Bhatt and F. T. Leighton, “A Framework for Solving VLSI Graph Layout Problems”. In *Journal of Computer and System Sciences*, v28, pages 300-343, (1984).

[4] P. Berman and G. Schnitger. “On the performance of the minimum degree ordering for Gaussian elimination” In *SIAM Journal of Matrix Analysis and Applications* v11(1), pages 83-88 (1990).

[5] H. L. Bodlaender, J. R. Gilbert,H. Hafsteinsson, and T. Kloks. “Approximating treewidth, pathwidth, frontsize, and shortest elimination tree”. In *Jornal of Algorithms* v18(2), pages 238-255, (1995)

[6] H. L. Bodlaender, J. S. Deogun, K. Jansen, T. Kloks, D. Kratsch, H. Muller and Z. Tuza, “Ranking of graphs” In *SIAM Journal on Discrete Mathematics* v11, pages 168-181, (1998).

[7] C. Bornstein, B. Maggs, G. Miller, and R. Ravi. “Parallelizing Elimination Orders with Linear Fill”. In *Proceedings of 38th Annual Conference on the Foundations of Computer Science*, (1997).

[8] S. N. Bhatt and F. T. Leighton, “A Framework for Solving VLSI Graph Layout Problems”. In *Journal of Computer and System Sciences*, v28, pages 300-343, (1984).

[9] E. Cuthill and J. McKee. “Reducing the bandwidth of sparse symmetric matrices” In *Proceedings of the 24th National Conference of the ACM*, pages 157-172, (1969).

[10] I. Duff, editor. *Sparse matrices and their uses*, Academic Press, (1981).

[11] I. Duff and J. K. Reid. *Direct methods for sparse matrices.* Oxford University Press, (1986).

[12] H. N. Djidjev. “Partitioning Graphs with costs and weights on vertices:algorithms and applications” *Lecture Notes in Computer Science* vol 1284, pages 130-143.

[13] H. N. Djidjev, S. M. Venkatesan. “Planarization of Graphs embedded on surfaces”. In *Proceedings of 21st Workshop on Graph theoretic concepts in Computer Science.* LNCS vol 1017, pages 62-72, (1995).

[14] G. Even, S. Guha, B. Schieber. “Approximation Algorithms for Graph drawing and VLSI layout problems”. To appear in *Symposium on Theory of Computing, 2000*

[15] K. A. Gallivan et al. *Parallel Algorithms for Matrix Computation.* SIAM, Philadelphia, PA (1990)

[16] N. Garg, H. Saran, and V. V. Vazirani. “ Finding Separator Cuts in Planar Graphs within Twice the Optimal”. *Proceedings of the 35nd Annual Conference on the Foundations of Computer Science*, pages 14-23, (1994)

[17] J. A. George. “Nested dissection of a regular finite element mesh” In *SIAM Journal on Numerical Analysis*, v10, pages 345-367, (1973)

[18] J. A. George. "An Automatic one-way dissection algorithm for irregular finite element problems". In *SIAM journal on Numerical Analysis*, v17, pages 740-751, (1980).

[19] J. A. George and J. W. Liu. "An automatic nested dissection algorithm for irregular finite element problems". In *SIAM Journal on Numerical Analysis* v15, pages 1053-1069, (1978).

[20] J. A. George and J. W. Liu. *Computer Solution of Large Sparse Positive Definite Systems.* Prentice Hall Inc. (1981).

[21] J. A. George and J. W. Liu. "The evolution of the minimum degree algorithm" In *SIAM Review* v 31, pages 1-19, (1989).

[22] N. E. .Gibbs, W. G. Poole, and P. K. Stockmeyer. "An algorithm for reducing bandwidth and profile of a sparse matrix". In *SIAM Journal on Numerical Analysis* v13, pages 236-250, (1976).

[23] J. R. Gilbert. "Some Nested Dissection Order is Nearly Optimal". In *Information Processing Letters*, v26, pages 325-328 (1987/88)

[24] J. R. Gilbert, D. J. Rose, and A. Edenbrandt. "A Separator Theorem for Chordal Graphs". *Siam Journal on Algebraic and Discrete Methods* v5, pages 306-313, (1984).

[25] J. R. Gilbert and R. E. Tarjan. "The Analysis of a Nested Dissection Algorithm". In *Numerische Mathematik* vol 50, pages 377-404, (1987).

[26] A. J. Hoffman, M. S. Martin, and D. J. Rose. "Complexity bounds for regular finite difference and finite element grids". In *SIAM Journal on Numerical Analysis*, v10, pages 364-369 (1973).

[27] B. Hendrickson and E. Rothberg. "Inproving the runtime and quality of nested dissection ordering". In *SIAM Journal on Scientific and Statistical Computing* v20(2) pages 468-489, (1998).

[28] J. Jess and H. Kees. "A data structure for parallel L/U decomposition". In *IEEE Transactions on Computers* v31, pages 231-239, (1982).

[29] U. Kjærulff. "Triangulations of graphs - Algorithms giving small total state space". R 90-09, Institute for Electronic Systems, Department of Mathematics and Computer Science, University of Aalborg, (1990).

[30] P. Klein, A. Agrawal, R. Ravi, and S. Rao. "Approximantion through multi-commodity flow". In *Proceedings of 31st Annual IEEE Conference on Foundations of Computer Science*, pages 726-737, (1990).

[31] F. T. Leighton and S. Rao. "Multicommodity Max-Flow Min-Cut Theorems and their Use in Designing Approximation Algorithms". To appear in *Journal of ACM.*

[32] C. E. Leiserson. "Area-efficient graph layouts (for VLSI)". In *Proceeding of 21st Annual Symposium on Foundations of Computer Science*, pages 270-281, (1980).

[33] C. E. Leiserson and J. Lewis. "Orderings for parallel sparse symmetric factorization". In *Parallel Processing for Scientific Computing*, G. Rodrigue, editor, pages 27-32, SIAM (1987).

[34] J. Lewis, B. Peyton, and A. Pothen. "A fast algorithm for reordering sparse matrices for parallel factorization". In *SIAM Journal on Scientific and Statistical Computing*, v10, pages 1156-1173, (1989).

[35] R. J. Lipton and R. E. Tarjan. "A separator theorem for planar graphs". In , *SIAM Journal on Applied Mathematics* v36(2) pages 177-189, (1979).

[36] R. J. Lipton, D. J. Rose and R. E. Tarjan. "Generalised nested dissection". In *SIAM Journal of Numerical Analysis*, v16 pages 346-358, (1979).

[37] J. W. Liu. "Modification of the minimum degree algorithm by multiple elimination" In *ACM Transactions on Mathematical Software* v12, pages 141-153, (1985).

[38] J. W. Liu. "The role of elimination trees in sparse factorization". In *SIAM Journal on Matrix Analysis and Applications* v11, pages 134-172, (1990).

[39] J. W. Liu. "Reordering sparse matrices for parallel elimination". In *Parallel Computing*, v11, pages 73-91, (1989).

[40] J. W. Liu and A. Mirzaian. "A linear reordering algorithm for parallel pivoting of chordal graphs". In *SIAM Journal on Discrete Mathematics*, v2, pages 100-107, (1989).

[41] J. W. Liu and A. H. Sherman. "Comparative analysis of Cuthill-McKee and the reverse Cuthill-McKee ordering algorithms for sparse matrices". In *SIAM Journal on Numerical Analysis*, v13, pages 198-213, (1976).

[42] H. M.Markowitz. "The elimination form of the inverse and its application to linear programming". In *Management Sciences* v3, pages 255-269, (1957).

[43] R. H. Mohring, "Graph problems related to gate matrix layout and and PLA folding" In *Computing Supplementum*, G. Tinhofer editor, v7, pages 17-51, Springer Verlag (1990).

[44] A. Natanzon, R. Shamir, and R. Sharan. "A polynomial Approximation Algorithm for the Minimum Fill-In Problem". In *Proceeding of Symposium on Theory of Computing* (1998).

[45] J. Nevins and D. Whitney editors, *Concurrent Design of Products and Processes*, McGraw–Hill, (1989)

[46] T. Ohtsuki, L. K. Cheung, and T. Fujisawa. "Minimal triangulation of a graph and optimal pivoting order in a sparse matrix." In *Journal of Mathematical Analysis and Applications*, v54, pages 622-633, (1976).

[47] S. Parter. "The use of linear graphs in gaussian elimination". In *SIAM Review*, v3, pages 364-369, (1961).

[48] A. Pothen. "The complexity of optimal elimination trees". Technical Report CS-88-16, Department of Computer Science, The Pennsylvania State University, University Park, PA (1988).

[49] D. J. Rose. "Triangulated Graphs and the elimination process". in *Journal of Mathematical Analysis and Applications*, v32, pages 597-609, (1970).

[50] D. J. Rose. " A graph theoretic study of the numerical solution of sparse positive definite systems of linear equations" in *Graph Theory and Computing*, R. C. Read editor, Academic Press, pages 183-217, (1972).

[51] R. Schreiber. "A new implementation of sparse Gaussian elimination". In *ACM Transactions on Mathematical Software*, v8(3), pages 256-276, (1982).

[52] A. Sen,H. Deng, and S. Guha. "On a graph partition problem with application to VLSI layout". In *Information Processing Letters*, v43, pages 87-94, (1992).

[53] F. Shahrokhi, O. Sýkora, L. A. Székely, and I. Vrťo, "Crossing Numbers: Bounds and Applications". In *Intuitive Geometry*, Bolyai Society Mathematical Studies 6, Budapest, pages 179-206, (1997).

[54] W. F. Tinney and J. W. Walker. "Direct solutions of sparse network equations by optimally ordered triangular factorization". In *IEEE Transactions on Computers* v29(7), pages 632-638, (1980).

[55] J. D. Ullman, *Computational Aspects of VLSI*, Computer Science Press, (1984).

[56] L. G. Valiant. "Universality Considerations in VLSI Circuits". In *IEEE Transactions on Computers*, vC-30(2), pages 135-140, (1981).

[57] M. Yannakakis, "Computing the Minimum Fill-in is NP-complete". In *SIAM Journal on Algebraic and Discrete Methods*, v2, pages 77-79, (1981).

A Chordal Completion Problems and Sparse Matrices

As an example for the framework of dissection, this paper addresses problems that arise in connection with sparse matrix computations. The most common such problem is finding a vector x satisfying the linear system $Ax = b$. This is useful for VLSI simulations, linear programming solution, signal processing (see [10]). One method of solving this is via Gaussian elimination process, in which a multiple of a row is subtracted from other rows. In this process if the row i is subtracted from row j (to eliminate the coefficient A_{ij}) and the entry A_{jk} becomes non-zero from being zero, it is defined as *fill*. This is unwanted, since if we started from a sparse matrix, the matrix becomes less sparse. The fill-in indicates the amount of memory we require to solve this system. Often these systems are quite large, which poses the question of efficient computation and it is prudent to choose a scheme which maximizes such. Several measures of efficiency are extant, the issues of minimizing the parallel dependency (elimination height), minimizing space requirements (chordal completion) and total work (operation count). There is a very large body of literature addressed to the problem of sparse matrix computation.

Most of the literature focuses on the case that A is a symmetric positive definite matrix, which allows the issues of numeric stability to be ignored. It also allows one to view the matrix as a graph, where the vertices are the columns (or the rows) are the vertices in the graph, with an edge from vertex i to j if the entry $A_{ij} = A_{ji}$ is non-zero. The Gaussian elimination process chooses a pivot i, and eliminates all entries A_{ji} with $j > i$. If two rows are independent with $A_{jk} = A_{kj} = 0$, then the operation of simultaneous pivoting is well defined and exploited by parallel solvers.

The elimination process on the graph essentially reduces to the removal of vertex i and transforming its neighbor set into a clique (because the corresponding entries in the matrix can become non-zero by this process). The entries which were zero previously and become non-zero in the process are termed as fill.

Parter in [47] interpreted the elimination process as a graph theoretic game, Rose in [49] demonstrated the connection of this problem with finding triangulated graphs. A graph is defined to be *strongly chordal* or just *chordal*, there exists no induced simple chordless cycle of length 4 or more. For such graphs it is easy to show that there exists *simplicial* vertices, such that its neighborhood is a clique. The family of chordal graphs is complete under subgraphs, and this essentially gives us an elimination ordering of zero fill. However if we consider any elimination order, the ordering will introduce a *chordal completion* of the given graph. Chordal completion also arise in context of pedigree analysis, evidence propagation in belief networks [29].

The fill-in is not the only measure of importance, the amount of work required to solve the linear system is sum of squares of all the cliques of the chordal graph constructed by the elimination process. Another measure of importance is the measure of how fast the system can be solved in parallel. This amounts to elimination of independent subsets in every step, and reduction of the number of such steps. However finding an ordering to minimize fill is NP-complete (see [57]), and similarly for minimizing elimination height (see [48, 5]).

However a large set of ordering heuristics have been developed, in [9, 11, 15, 17, 18, 19, 20, 21, 22, 37, 41, 42, 46, 50, 54]. Most of these approaches use one of two methods, using minimum degree heuristic (see [37, 42, 54] for example) and nested dissection (see [17, 19, 36]) to reduce the fill-in operation count or the elimination height. Hybrid algorithms (see [2, 27] for example), are usually considered the best.

Compared to the attention received by fill-in problem, the operation count problem has not been studied that extensively. Some results are shown in [26, 36, 25]. Minimizing the elimination height has been studied with regard to eliminating more than one row in parallel. The elimination tree was first defined in [51], although several algorithms prior to this paper used the idea. Considerable number of other heuristics have been proposed for this problem as well, see [15, 28, 33, 34, 39, 40].In [38] an excellent survey on elimination tree and its uses is presented. In [28] the problem was studied for chordal graphs, which was proven to be optimal in [39], otherwise no performance guarantees were studied for this problem.

In [1, 30] the first polynomial time approximation for these proposed. This was improved and supplemented in [44]. In [1] it was shown that there exists an ordering which simultaneously approximates the three most known parameters, and they also provided approximation results to this end. In our work we improve both the existential and the approximability results. In [7] it was investigated if there exists better orderings for chordal graphs to expose parallelism, increasing the fill or work required by a small amount.

B Two way or Three way

Definition 11 *A super-additive estimator ϕ is defined to* **special** *if for two arbitrary subsets P and Q of the vertex of the graph G, (we abuse the notation to denote the induced subgraphs by the subsets) the following is satisfied,*

$$\phi(G-P)+\phi(G-Q)+\phi(P\cup Q)\leq 2\phi(G) \tag{1}$$

It is easy to observe that equation 1 implies super-additivity (set $P=G-Q$). It is also easy to verify that an estimator which is the number of edges or vertices satisfy equation 1.

Lemma B.1 *Any estimator, which can be expressed as an assignment of weights on vertices and edges and subsequent sum over them with the property that these weights do not increase under subgraph relation, (the function will not assign larger weights to vertices and edges of an induced subgraph) will satisfy equation 1.*

Proof: The crucial idea is that the weight of $\phi(G-P)$ is at most the weight of the subgraph of G^* induced by $(G-P)$. Likewise for the other subgraphs on the left hand side.

Now we use the graph G^* and its subgraphs for both sides of the equation. The proof follows from tracing the contribution of every vertex or edge to both sides of the equation. A vertex or edge appears at most twice on either side and in the left side its contribution is smaller by assumption. □

We can observe that the crossing number of a graph and the minimum fill-in number are special super-additive estimators.

Theorem B.1 *For a special super-additive estimator $\phi()$, in polynomial time a two way decomposition tree can be constructed where the parameter t at the root of a subtree (corresponding to a subgraph G') decreases by a factor of $2/3$ along a path, and the separator corresponding node is at most $O(\alpha(G')f(t,G'))$.*

It will not be very instructive to repeat the whole algorithm, however we will indicate the changes. We will make all subcalls of $\textsc{Partition}(t,G)$ with the value of the first parameter $2t/3$ (instead of $t/2$). And the last step when A and B both return success we change the substeps $\{i, ii, iii\}$, and instead run the following, (the other two steps remain as before).

i We now try $\textsc{Partition}(2t/3, G - A - S_i)$ and $\textsc{Partition}(2t/3, G - B - S_i)$.

ii If both of the above were failure, we report failure.

iii If the former succeeds, we construct $A - S_i - X_i$ and $G - A - S_i$ as the partitions (with their respective decompositions) and $S_i \cup (A \cap X_i)$ as the separator. In case of latter we construct $B - S_i - X_i$ and $G - B - S_i$, with separator $S_i \cup (B \cap X_i)$.

All the conclusions as the last section ensue. To see the correctness, the argument will follow the exact lines of lemma 3.1.

The the only interesting case to consider is the case when both A and B were successfully partitioned, but $\textsc{Partition}(2t/3, G-A-S_i)$ and $\textsc{Partition}(2t/3, G-B-S_i)$ returned failure. This tells that $\phi(G - A - S - i)$ and $\phi(G - B - S_i)$ are both greater than $2t/3$.

Clearly $i \neq 0$, since in that case $B = G - A - S_0$. Now for the graph N_{i-1} itself, there must have been a call to $\textsc{Partition}(2t/3, N_{i-1})$ at the i'th iteration which must have been a failure. This tells us that $\phi(N_{i-1}) \geq 2t/3$. Since $N_{i-1} = A \cup B \cup S_i$, setting $P = A \cup S_i$, and $Q = B \cup S_i$, we get

$$\begin{aligned} 2\phi(G) &\geq \phi(G-P) + \phi(G-Q) + \phi(P \cup Q) \\ &= \phi(G - A - S_i) + \phi(G - B - S_i) + \phi(N_i) > 2t \end{aligned}$$

This is a contradiction.

The proof of polynomial running time almost remains the same, with the constant $2/3$ replacing all occurrences of $1/2$. This just increases the exponent, and the same line of proof goes through.

C Nested Dissection with Several Estimators

We will not fully state this algorithm since the algorithm will proceed as in $\textsc{Partition}(t,G)$, we will merely indicate the changes required. Denote the current algorithm by $\textsc{Partition}(t_1, t_2, \ldots, t_p, G, r)$. The index r will cycle over the indices $1, 2, \ldots, p, 1, \ldots$ and keep track of which estimator to partition the graph with respect to.

The algorithm will essentially be the same as $\textsc{Partition}(t,G)$, except that all recursive calls of $\textsc{Partition}(t_1, t_2, \ldots, t_p, G, r)$ will involve $t_r/2$ and $r \ (mod\ p) + 1$. All the separators constructed will be tested against all the functions $f_1, f_2, \ldots, f_p$.

The correctness will follow from exactly the same lemma as in lemma 3.1 above. The proof of polytime is also analogous to lemma 3.1, the recurrence $g(t_1, t_2, \ldots, t_p, n, r)$ can be written as

$$g(t_1, t_2, \ldots, t_p, n, r) \leq p(n) \quad + \quad g(t_1, t_2, \ldots, t_i/2, \ldots, t_p, (1-a)n, r+1)$$

$$
\begin{aligned}
&+ \sum_{j=1} g(t_1, t_2, \ldots, t_r/2, \ldots, t_p, a^j n, r+1) \\
&+ \sum_{j=1} g(t_1, t_2, \ldots, t_r/2, \ldots, t_p, n, r+1) \\
&+ \sum_{j=1} g(t_1, t_2, \ldots, t_r/2^j, \ldots, t_p, n, r+1)
\end{aligned}
$$

with $r+1$ denotes $r\ (mod\ p)+1$ and a is the fraction of the vertices of V_i in the larger component after the separation. The basic essence of the proof will be that from one invocation of PARTITION() to the next, the quantity $n\prod_j t_j$ decreases by a constant factor. We can now claim the following theorem,

Theorem *If $\phi_1, \phi_2, \ldots, \phi_p$ are super-additive estimators with functions $f_1, f_2, \ldots, f_p$ which relate them to the separator size with respect to arbitrary subsets, in polynomial time a decomposition tree can be constructed such that the separator for G' has size at most $\alpha(G')f_j(s_j, G')$ for all $j \in \{1, \ldots, p\}$ and subgraphs G' induced by any subtree of the separator tree. Furthermore in every p'th level starting from level j, the guess s_j for the estimator ϕ_j decreases by a factor of $1/2$ along any path in this tree.*

D Proofs in Section 4

Lemma *Given a tree with weights on vertices, such that the weights along any path from root to a leaf decrease by a factor of $\gamma < 1$ and or every anti-chain of size s the inequality $\sum_i y_i \leq s^a W$ holds, where W is the weight of the root, and a, γ are constants strictly less than 1; then the sum of the vertex weights along any r paths can be bounded by $O(r^a W)$.*

Proof: Consider the subtree spanned by these paths. From each leaf, we walk upwards till we hit a vertex of degree three. We consider the paths (excluding the vertex of degree three). It is trivial that the roots of these paths are anti-chain, since otherwise some node on some path would have had degree three. Thus the sum of the roots of these paths have weight at most $r^a W$. Since the weights decrease geometrically, the weight of each path is at most $1/(1-\gamma)$ times the weight of the root. Thus the total weight considered is at most $r^a W/(1-\gamma)$.

Now consider the tree with these paths deleted. We have a similar tree with at most $r/2$ leaves. Since at least two of the original leaves are needed to generate a leaf in the modified tree. We now carry on the process mentioned above repeatedly, which shows that the total weight is at most $r^a W/((1-\gamma)(1-2^a))$, which proves the lemma. □

Theorem *The above algorithm gives an $O(m^{\frac{1}{2}} log^{\frac{3}{2}} n/\phi(G)^{\frac{1}{4}})$ approximation for the chordal completion for arbitrary graphs.*

Proof: Assume that we are counting the contribution of the nodes in separator X_i whose descendents induce the subgraph G'. By lemma 4.3 the contribution of node v is $O(\sqrt{d_v \phi(G')} \log n)$, where d_v denotes the number of *downward going* edges of v in the tree decomposition. Let $down(G')$ denote the number of downward edges in the at the root of the subtree corresponding to dissection of G'. By application of Holder's inequality we have $\sum_v \sqrt{d_v}$ bounded by $\sqrt{|X_i|}\sqrt{\sum_v d_v}$. The last term $\sum_v d_v$ is the number of downward going edges of G' and is $down(G')$. If we now consider the contribution from all nodes at the same level j in the separator tree, if $||X||_j$ denote the largest separator, the contribution is $O(||X||_j \sum_{G'} \sqrt{down(G')}\sqrt{\phi(G')} \log n)$. Once again, applying Holder's inequality, this is bounded above by $O(||X||_j \sqrt{\sum_{G'} down(G')}\sqrt{\sum_{G'} \phi(G')} \log n)$. Due to super-additivity, the term

$\sum_{G'} \phi(G')$ is at most $\phi(G)$ and the term $\sum_{G'} down(G')$ is at most m, which is the total number of edges in the graph. Thus the contribution from the level j is at most $O(||X||_j \sqrt{m\phi(G)} \log n)$. Now the value $||X||_j$ is bounded above by $O(\sqrt{\phi(G'')} \log n)$ for the subgraph G'' with the largest $\phi()$ value. Since the $\phi()$ decreases geometrically summing the sum $\sum_j ||X||_j$ is at most $O(\phi(G)^{\frac{1}{4}} \log^{\frac{1}{2}} n)$. This completes the proof. □

E Proofs in Section 5

Lemma *The minimum operation count, OPT_m, is a special super-additive estimator.*

Proof: To see this, let us consider the elimination ordering $\beta()$ for the graph G which minimizes the number of operations. This ordering fixes a chordal extension of G, say to G^*. As the elimination proceeds, each operation corresponds to an edge of G^*. Thus for each operation we can charge an edge of G^*, the total charge being the number of common neighbors of the endpoint plus one.

Now consider the same elimination order restricted to any subgraph. The charge introduced by $\beta()$, on the edges of the subgraph is at most the charge introduced on G^*. The proof follows by the application of lemma B.1. □

Lemma *If the minimum number of multiplications required by a graph is OPT_m, then for any arbitrary set S there exists a separator of size $O(\sqrt[3]{OPT_m})$.*

Proof: Consider the chordal extension of the graph produced by the elimination ordering, and the weight function introduced by vertices of S having weight 1 and others 0. The smallest separator, since the graph is chordal, has to be a clique. Since the smallest separator is a clique, the number of operations have to be at least cube of the size of the separator under any elimination order. This proves the lemma. □

Lemma *Consider a node v and its associated node X_v in the separator tree, and the graph induced by all vertices in the descendent nodes of X_v be G'. The charge at v is at most $O(d^{2/3} OPT_m(G') \log^3 n)$.*

Proof: Consider an edge (w, v). The charge it will acquire is all the common neighbors of v and w created over the entire process. Denote by G_w the subgraph induced by all nodes descendent of X_w. Assume that the number of neighbors of v in the separator rooted at X_w is at most y_w. Now over an anti-chain, the cube-roots of the estimator are super-additive, and thus the sum of s cube-roots of the estimators on anti-chain nodes is at most $s^{2/3}$ times the root. This follows from a simple application of Holder's Inequality. Thus the number of mutual neighbors of v and w are at most $O(y_w^{2/3} OPT_m(G_w)^{1/3} \log n)$. Therefore summing over all node in X_v, this is at most $O((y_w OPT_m(G_w))^{2/3} \log^2 n)$. We apply Holder's Inequality again,

$$\begin{aligned}
\sum_w y_w^{\frac{2}{3}} OPT_m(G_w)^{\frac{2}{3}} &\leq \left(\sum_w (y_w^{\frac{2}{3}})^3 \right)^{\frac{1}{3}} \left(\sum_w (OPT_m(G_w)^{\frac{2}{3}})^{\frac{3}{2}} \right)^{\frac{2}{3}} \\
&\leq (\sum_w y_w^2)^{\frac{1}{3}} (\sum_w OPT_m(G_w))^{\frac{2}{3}} \leq (\sum_w y_w)^{\frac{2}{3}} OPT_m(G')^{\frac{2}{3}}
\end{aligned}$$

This totals to $(dOPT_m(G))^{2/3}$. Thus the net contribution from all all vertices in X_v is at most $O(d^{2/3} OPT_m(G') \log^3 n)$. □

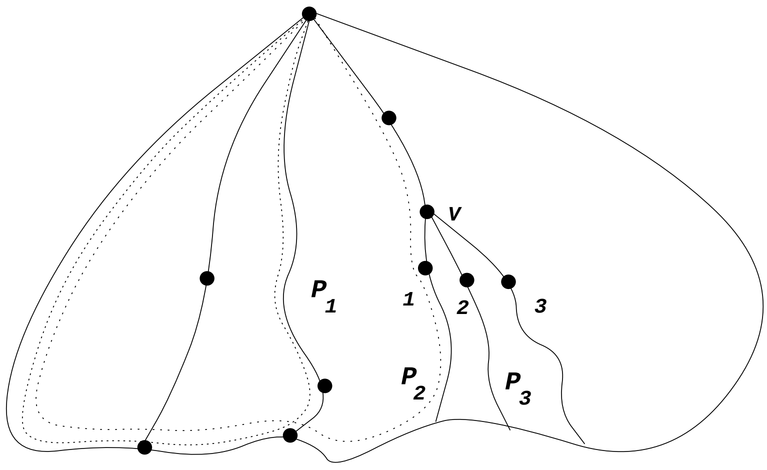

Figure 2: Laminar cuts introduced by the paths

F Proofs in Section 6

Lemma *The elimination height for chordal graphs can be approximated for chordal graphs up to a factor of $O(\log n)$.*

Proof: The proof is very straightforward, based on the fact that we can find balanced clique separators for the chordal graph. Now, since the vertices in the separator form a clique, the elimination height of any ordering (which may possibly introduce new edges) has to be at least the size of the separator. And trivially, the elimination heights of the subgraphs produced are at most the height for the original graph. Since there are $O(\log n)$ levels in such a decomposition, the elimination height achieved by such a separator tree is at most $O(\log n)$ times the optimal. □

Lemma *If the elimination height of a graph is H, and its minimum fill-in k, then there exists a set of size $2|H|$, whose deletion disconnects the graph in two parts such that the fill-in of both parts is at most $2k/3$. In this sense this is a separator (with respect to fill-in and not the number of vertices or edges).*

Proof: Consider the tree which has elimination height H. This tree has a single vertex of the original graph at every node and the children nodes can be eliminated in parallel. Consider the set of paths from the root to the leaves. These paths can be ordered in a natural fashion. Number the children nodes at a node of this tree in some order. The paths which take a lower numbered branch at this node are before the paths which take a higher numbered branch.

In figure F, for example the path P_2 is before the path P_3, since the latter passes through child number 2 of v and the former through 1. Each of these paths, disconnect the graph on deletion, since there cannot be an edge between the subtrees that hang from this path. This is because if there were an edge, then that edge must enforce that one endpoint is ancestor of the other and hence is not possible.

It is easy to observe that these paths introduce cuts, and these cuts are laminar. For example all the vertices on the left of the path, if the paths were drawn on a plane (see figure F). So either there exists a path on whose deletion both parts have fill-in at most $2k/3$ (one part can be empty) or there there exists two consecutive paths, such that the smaller order path introduces a piece with fill-in at most $k/3$ and the other piece has fill-in $2k/3$ and removal of the second path from this larger piece results a graph with fill-in $k/3$. This proves the lemma. □

Corollary *If the elimination height of a graph is H, then by deleting H vertices we can bisect an arbitrary set in the graph, such that each part has at most $2/3$ fraction of the vertices.*

Proof: We consider the tree defined by the smallest height elimination ordering, The laminar set of cuts allows us to prove this result, which is stronger than the above lemma since vertices are unweighted. □

G Proofs in Section 7

Theorem *If the minimum elimination height is H, minimum fill-in F (not necessarily achieved by the same ordering) and minimum number of operations M, there exists a nested dissection ordering which has height $O(H \log n)$, fill $O(\sqrt{d}(m+F)\log n)$ and number of operations $O(d^{2/3} M \log n)$.*

Proof: We decompose the graph in the following fashion, in odd levels we partition the graph with respect to the chordal completion size, in the even levels we separate the graph with respect to the number of operations. We use the smallest separator to achieve this.

The proof follows from the observation that the ordering corresponding to the minimum elimination order allows us to construct separators of any weighted set of size at most $2H$. Thus over $O(\log n)$ levels, the first part of the result follows.

For the case of fill-in, we assume that a graph G' is split into four pieces while paying for three separators, each of which are at most the square-root of the minimal chordal completion size of G'. This also uses the observation that any weighted set can be separated by these many vertices. In this case the weights on the vertices are defined in a bit more complicated fashion. Each edge in the chordal super graph corresponding to the minimum number of operations gets a charge, which is the number of vertices adjacent to both ends. Now each vertex gets half the sum of the charges on the edges incident on it. So if this weighted graph is separated, then the number of operations in either side will decrease by a constant. Now by [24], this is achieved by a set of size at most the square-root of the number of edges plus fill-in. In the odd levels, this fact is true if the weight function is defined by each vertex having a weight of half its degree in the minimum chordal extension.

Observing the above, as in the previous section, the separator sizes decrease geometrically (We are only considering the odd levels and the separators in the even level are bounded by the upper bound of the their parent separator.) and this allows us to conclude that the number of edges are at most $O(\sqrt{d}(m+F)\log n)$.

Similarly, considering the even levels bounds the number of operations to be at most $O(d^{\frac{2}{3}} M \log n)$. □

Theorem *If the minimum elimination height is H, minimum fill-in F and minimum number of operations M, then in polynomial time we can construct a nested dissection ordering which has height $O(H \log n \log F)$, fill-in $O(\sqrt{d}(m+F)\log^2 n \log F)$ and number of operations $O(d^{2/3} M \log^3 n \log F)$.*

Proof: We use the three estimators, the fill-in, minimum chordal completion size and the number of operations. (Clearly the second one can be replaced by the sum of edges and the fill-in, however for simplicity of presentation, we will stick with three estimators.) We use the three functions, $f_1()$ which is a constant, $f_2()$ which is square-root of the chordal completion size and $f_3()$ which is cube-root of the number of operations.

We use the fact that in each step of the iteration we are interested in constructing minimum separator with respect to vertex subsets. And we obtain a $O(\log n)$ approximation of the minimum separator. Clearly if the minimum elimination height is H, the minimum separator can always be bounded by H. (Note that the algorithm does not use weighted subsets, so there exists an exact $1/2$ separator of size at most H.) The number of levels where the graph is not chordal and we have separators of size $O(H \log n)$ is $O(\log F)$, and after this phase, all the clique separators we find are at most H. Thus the elimination height of the dissection we construct is at most $O(\log F \log n)$ times H.

Consider the dissection steps where we dissected with respect to the completion size. If the current graph is G', we know that over the next two levels (six separators) the minimum separators are of size at most $O(\sqrt{2\phi(G')/3})$, where $\phi(G')$ is the minimum completion size. This is obvious if we consider the clique separators introduced by the ordering corresponding to the minimum fill-in for G'. We achieve a $O(\log n)$ approximation for these. The eight pieces of G' constructed at the third level are all guaranteed to have completion size at most $2\phi(G')/3$. Thus we can conceptually treat all the seven separators (the one splitting G' and the six separators in the two levels below) as a single separator, and once again we are in a situation that the upper bound on the separators constructed decreases by a constant factor. This immediately shows that the number of edges introduced due to the top $O(\log n)$ levels is at most $O(\sqrt{d}(m+F)\log^2 n \log F)$. Over the levels when the graph is chordal, the graph is split according to completion and operation count only. Except that we construct minimal clique separators. These cliques are also embedded in any ordering that minimizes fill-in. Therefore for these levels we find the minimum separator and the contribution is $O(\sqrt{d}(m+F)\log n)$, which is absorbed by the previous term.

The same analysis carries over for the minimum operation count. If we consider the ordering that induces such, for G' we have separators of size $O(\sqrt[3]{OPT_m(G')}$. The six separators in the two next levels below this separator have separators at most of size $O(\sqrt[3]{2OPT_m(G')/3})$. Thus over every third level the upper bound on the separator decreases by a constant factor and we achieve an operation count of $O(d^{\frac{2}{3}} M \log^3 n \log F)$, once again the top $O(\log F)$ having the dominant contribution. □

H Minimum Crossing Number and Drawing Size

In this section we show the connection of our method to graph completion problems, where the intended completion family is planar graphs instead of chordal graphs.

The *Crossing Number Problem* is defined as follows, given a graph G, to provide an embedding of the G in the plane mapping vertices to points and edges to simple curves, such that the number of edges which intersect is minimized. At most two edges can intersect at a *crossing* point other than a vertex.

The *Minimum Planar Completion Problem* is the same problem with the objective to minimize the total number of points (vertices and crossing points). This problem is useful in considering layout problems.

The *Minimum Area VLSI Layout* of a graph is defined as, given a graph G of degree at most 4, to embed the vertices in a grid of smallest area such that the edges are mapped to paths on the grid which do not share any edges (of the grid).

In [14] the basic idea of estimators and geometrically decreasing separators were applied to approximate the minimum drawing size and the layout area problems. In [32] it was shown that planar graphs can be embedded with area $O(n \log^2 n)$. Following [31] it was known that the minimum drawing size can be approximated to a factor of $O(d^2 \log^4 n)$ for graphs of bounded degree d. This gave a $O(\log^6 n)$ approximation for layout area. These ratios were improved to $O(d^2 \log^3 n)$ and $O(\log^4 n)$ in [14]. Our general framework, subsumes these results and specially the simultaneous approximation, allows to show us improved result for a graph with crossing number k, by approximating the minimum drawing size within $O(d^2 \log^2 n \log k)$.

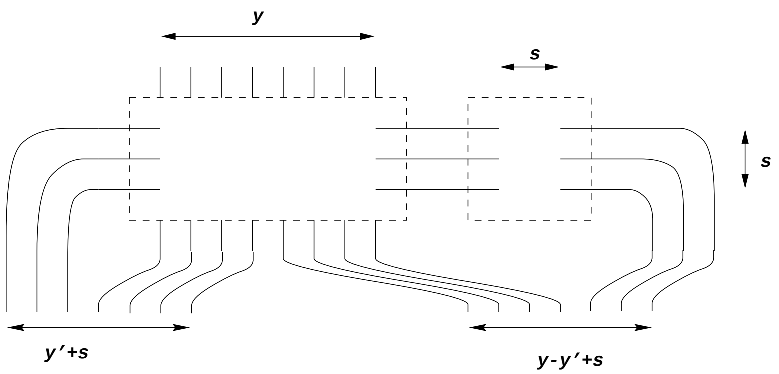

Figure 3: Drawing edges using separator tree

To see this, apply the simultaneous decomposition on the two estimators of and total number of points (drawing size) and crossing number in alternate levels. We observe that there exists small separators of size at most the square root of the drawing size which partitions an arbitrary set. The two functions f_1 and f_2 are chosen in an obvious fashion. After $O(\log k)$ levels we get planar graphs.

At this point we ignore the first estimator and separate with respect to the second parameter. There is a different way of getting this result, since k is small and a simple separator will achieve balance with respect to the number of points and crossing number. However it is simpler to state the result using theorem 3.2.

It is quite simple to observe that the separator tree can be used to embed the graph. The first step is to convert the vertex separators into edge separators. We delete all the edges incident on the separating vertices and divide the isolated vertices in the two parts of the tree. Thus the tree constructed has all the original vertices of the graph at the leaf nodes. We order these leaf vertices in a post order traversal of the entire tree. If there are more than two vertices at a leaf we order them arbitrarily. Thus we have a numbering on the vertices of the graph. Now we will show that the number of crossings required is similar to the product of separator calculation of the chordal completion size.

First consider a (edge) separator of size s, and the edges descending this separator subtree (the edges which separate the induced subgraph from the rest in the original graph) be y. Consider a gadget as shown in figure 3. We first assert that the s edges which appear on the left side of the tree can be in an arbitrary order (depending on the order of their endpoints) and similarly in the right side. Assume that we have two orderings now and we wish to connect a permutation (represented by the edges) amongst the orderings. Thus if we construct an $s \times s$ grid, we can use the grid to connect the two orderings. If the first edge on the left side appears on the other side as the ℓ'th edge, we use one row 1 and column s (we stop when we reach ℓ'th row on this column). Then we have an $s-1$ size permutation to deal with and we have an $(s-1) \times (s-1)$ grid. Thus the s edges are taken care of. Now the y descending edges are already in order, by assumption. But the s edges may interleave them in any order. By the same ideas as as used in the $s \times s$ grid, we can construct a $y \times s$ grid and connect all the edges, see figure 3.

Thus by the above the number of points in our drawing is $(s+y)s$ over all separators of size s. However this counting is quite difficult to handle. Consider the alternate way to count the same quantity. Observe that an edge at separator at the root of a subtree will cut all the edges along the path traced by the two endpoints. But this is the sum of the separators along these two paths, which we have seen before (and the geometric upper bounds have helped us). Thus the number of points can be counted by charging each separator twice the product of itself and the maximum sum of separators along any path to the leaves[6].

In this case, it is trivial to observe that if we construct the decomposition tree with respect to the two estimators, the result for the drawing size improves. The contribution over the top $O(\log k)$ levels amounts to $O((n+k)\log^2 n \log k)$ points per level. This is because each separator is at most $O(\sqrt{n+k}\log n)$ and each edge in the separator cuts at most all the edges associated with separators along the path taken by the edge. (Since one end point of the edge is in the separator set, the other endpoint will be in some descendent separator node.) Using the geometrically decreasing property, and using the bounds for the even levels by their parent separator (in the odd level we separate with respect to drawing size), the number of edges associated with separators along any path is at most $O(d\sqrt{n+k}\log n)$. After that the edge cuts at most $O(d\sqrt{n})$ edges in the levels where the subgraphs are planar. We use the same arguments as in the previous sections, of level by level accounting and super-additivity. Therefore we have the following theorem,

Theorem *The minimum drawing size problem can be approximated up to a factor of $O(d^2 \log^2 n \log k)$.*

Notice that this immediately proves the $O(d^2\log^3 n)$ result proved in [14]. We can make the stronger statement that the graph can be embedded in a $\sqrt{2/3}$-bifurcator (see [8]). However it is not apparent if the layout area to be approximated better than $O(\log^4 n)$ as shown in [14], to $O(\log^2 n \log^2 k)$ for example. The main difficulty appears to be the fact that since the dissection produces subgraphs with varying sizes. The subgraphs have to be packed in a different order since the analysis in [32, 14] can handle only comparable sizes to pack together.

H.1 Planarizing Sets and $o(\log^2 n)$ Crossing number

In this case we can use the result of [13], which proves the following theorem,

Theorem H.1 *[13] Graphs of genus g have a set of vertices of size $\sqrt{gn\log g}$ which can be found in linear time, whose removal yields a planar graph.*

Such a set is defined to be a *Planarizing Set.* It is a trivial observation that if the crossing number is k, then the graph can be embedded in a graph of genus k. Thus we use the above *planarizer* to get a planar graph and decompose the planar graph recursively. This is required since the vertices removed will require edges drawn to the vertices which are possibly in the interior of the planar subgraphs. We can easily observe the following theorem,

Theorem *The above decomposition tree draws a graph with $O(n(k\log k + \log n))$ points, which is $O(\max(k\log k, \log n))$ approximation for the minimum drawing size problem.*

For the layout problem we use the embedding in [32] and in the last step, add the requisite number of rows and columns to connect the edges in the *planarizing set.* Thus we get the result,

Theorem *The VLSI layout problem can be approximated within a factor of $O(\max(log^2 n, k\log k))$.*

[6]The separator itself is added to the path, notice that otherwise the $s \times s$ grid is unaccounted for.

I Planar drawings, Pathwidth and Cutwidth

Almost planar graphs have small pathwidth, and therefore small elimination tree height. In this section we explore the connections between fill-in, crossings and pathwidth. The following theorem is immediate,

Theorem *The pathwidth of a graph can be approximated within $O(\log n \log k)$, where k is the minimum fill-in.*

Proof: The proof follows from the same analysis as in the approximation of elimination height. The central observation is that if the pathwidth is W, then any subset can be separated using W vertices. $\square$

At this point we demonstrate an interesting connection between pathwidth and crossing number,

Theorem *The pathwidth can be approximated within factor $O(\log k \log n)$ for a graph with crossing number k.*

Proof: We essentially run the same algorithm with crossing number as the estimator and a constant $f()$, till the subgraphs have crossing number 1 or are planar. For the graphs with crossing number 1, we can guess which vertex to remove to make it planar. After this point we use a constant factor approximation for the separator as in [16]. Thus over $O(\log n)$ levels while we are decomposing planar graphs, we have a pathwidth $O(W \log n)$. As in the previous theorem this gives us a $O(\log n \log k)$ approximation for the pathwidth. $\square$

These results extend to cutwidth for bounded degree graphs,

Corollary I.1 *The cutwidth for graphs with bounded degree can be extended to $O(\log n \log k)$ where k is the fill-in or the crossing number.*

I.1 Gaussian Elimination on almost Planar Graphs

In [25] it was shown that planar graphs require $O(n \log n)$ fill and at most $O(n^{\frac{3}{2}})$ operation count. We can observe the following for graphs with small crossing number (and degree bound d),

Theorem I.2 *For a graph with crossing number k, we can find a dissection in polynomial time with fill-in $O(\sqrt{d}(n+k)\log^2 n \log k)$ and operation count $O(d^{\frac{2}{3}}(n+k)^{\frac{3}{2}} \log^3 n \log k)$ in polynomial time while approximating the elimination height up to a factor of $O(\log n \log k)$ of the best possible.*

The result follows using simultaneous approximation with the crossing number and total number of points. At each step the graph is tested to be planar. Once the subgraphs are planar (or crossing number is 1) we can use constant factor approximation of the separator along the lines of [16]. For the results on fill-in and operation count we can also use the algorithm of [25], however it is not apparent that it will achieve the requisite bounds on elimination height. The use of planarizing sets also yields some results in this context, but unfortunately most of these results are weak. It is not obvious how to extend the techniques used in this paper to get stronger results.